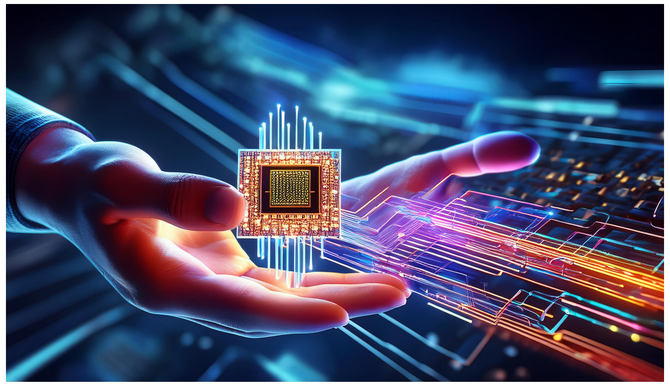
1/5/26 09:54

# CSCI 250
# Introduction to Computer Organisation
# Lecture 2: Computer Memory I



Jetic Gū

# Overview

- Focus: Course Introduction
- Architecture: Logical Circuits
- Textbook: v4: 8.1, 8.2
- Core Ideas:
  1. Random Access Memories (Hardware)
  2. Random Access Memories (Software)
  3. SRAM

# Random Access Memories (Hardware)

P1
RAM

# So far, in Storage

- Registers: limited in numbers, usually less than a few dozen in a single CPU (core)
  - Datapath has direct access to all GPRs
  - Read: Multiplexer; Write: Decoder
- Storage device: SSD, HDD, CD-ROM, Flash drives, etc.
  - Datapath does not have direct access to these
  - Data buses, device specific controllers, etc.

Review

# Two Types of Memory

- Here, we are talking about Memory in the von Neumann architecture sense
- Read Only Memory (ROM), non-volatile
  - Traditionally, Read-Only; Modern ROM are sometimes rewritable through special means
  - Provide storage for information that doesn’t need to be changed
  - e.g. Firmware

Concept

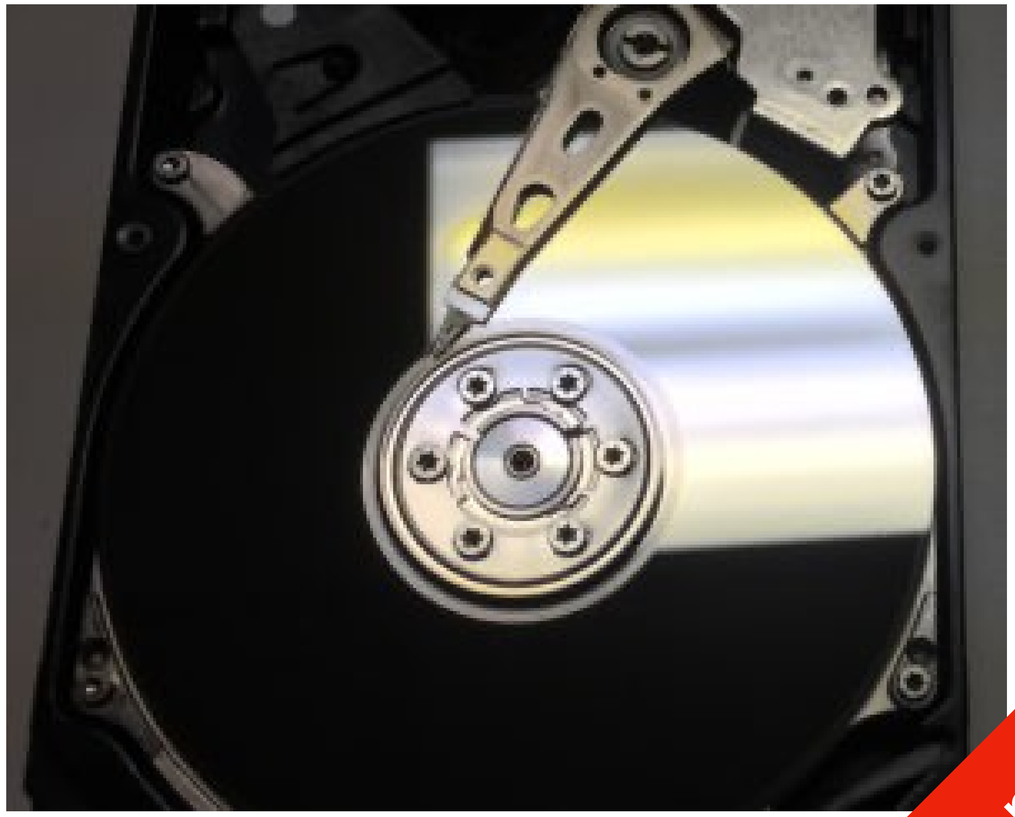
# Two Types of Memory

- Here, we are talking about Memory in the von Neumann architecture sense
- Random Access Memory (RAM): volatile
  - Read/Write operations
  - Similar to registers, but with different implementation technologies (slower)
  - CPU provides address and access mode, and after delay, information is accessed

Concept

# Why Random Access

- Non-volatile Hard Drives, Cassettes, Tapes (Serial Memory)
  - Sequential Access: data block $i$ is adjacent to data block $i + 1$
  - Seek operation: locate data block $i$
  - Seek first (slow), then sequential access (relatively fast)
- RAM: much much much faster than HDD, slower than registers (can be as fast but very expensive), doesn't require seeking



Concept

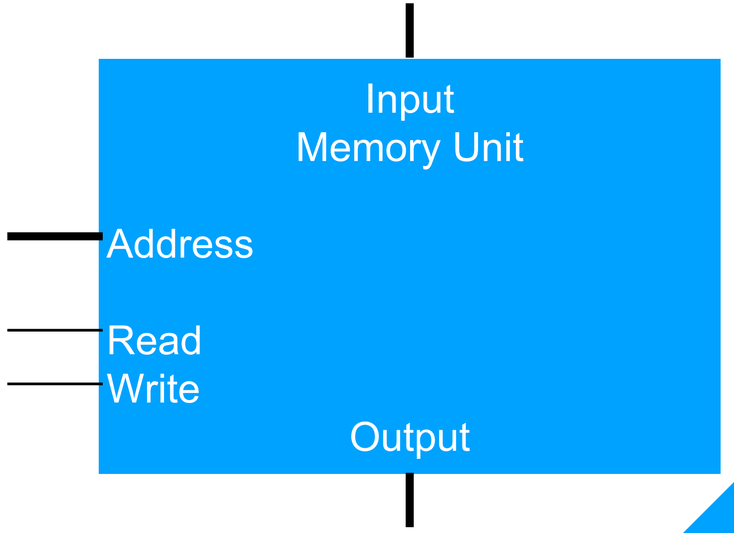
# RAM

- Word: the natural unit of data used by a particular processor design (processor dependent)
- RAM spec (processor independent)
    1. **number of bits per address**, most RAMs are byte addressable (1 byte per address);
    2. number of bits **accessed at read/write operation**
    3. number of **addresses** in the RAM unit
- Every word in your RAM can be accessed through an **address**

Concept

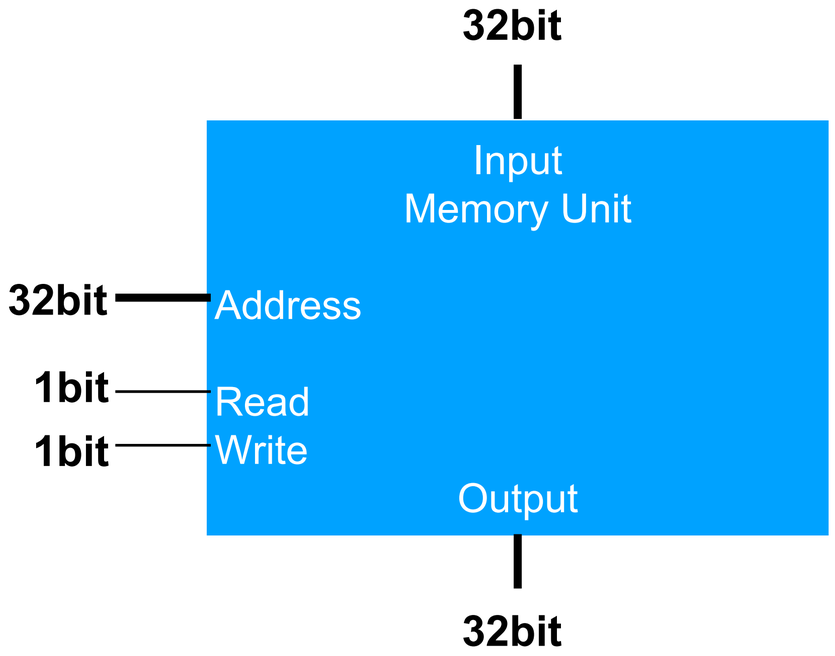
# RAM

| | Word Length | Bits for addressing | Bits per access | Max Addressable Memory |
|---|---|---|---|---|
| **x86 (8086/8088)** | 16bit | 20bit | 16bit | $2^{20}$ = 1MB |
| **x86-64** | 16bit | 64bit | 64bit | $2^{64}$ |
| **ARM32** | 32bit | 32bit | 32bit | $2^{32}$ = 4GB |
| **ARM64** | 32bit | 64bit | 64bit | $2^{64}$ |

# Byte Addressable RAM

- e.g. ARM32
  - Each byte has an address
  - Every access is performed on 32bits
  - e.g. read 00000000h -> 00000000h & 00000001h & 00000002h & 00000003h






1. https://developer.arm.com/documentation/107565/0101/Memory-system/Memory-address-space?lang=en

# Speed Comparison

- Your motherboard has at least one clock
  - crystal oscillator: the clock on which most other devices' speed is based
  - Your CPU: e.g. 3.5x
  - Your memory: e.g. 1x
  - PCIe bus: e.g. 0.5x
  - etc.

# Speed Comparison

- Rough estimation here for RAM, SSD, HDD, dependent on implementation technologies, processor generation, clock speed, and other factors
  - Registers: as fast as CPU itself
  - RAM: ~100x slower than registers (doesn’t match clock speed difference, why?)
  - SSD: ~5x-10x slower than RAM (has latency for each access request)
  - HDD: ~1x + slower than SSD (has extra latency for each seek)

# Computer RAM technologies

- Static RAM (SRAM): uses latches
  Fast, so expensive

- Dynamic RAM (DRAM): uses capacitor + transistor
  Much slower, but not as expensive
  requires refreshing, not strictly synchronous

- Synchronous Dynamic RAM (SDRAM)
  DRAM, but synchronous

P1
RAM

# Computer RAM technologies

- Single Data Rate Synchronous Dynamic RAM (**SDR SDRAM**)
  added mode register, load/store takes multiple cycles, but CPU can track progress using the mode register

- Double Data Rate Synchronous Dynamic RAM (**DDR SDRAM, DDR2, DDR3, DDR4**)
  Positive edge and Negative edge both output different pieces of info, so double the data rate

- Graphics Double Data Rate Synchronous Dynamic RAM (GDDR SDRAM, GDDR2, GDDR3, GDDR4, GDDR5)

- Flash Memory: limited life span, every write costs life

# Random Access Memories (Software)

# A Programmer's Perspective

- When you write a programme, usually your code is not directly accessing physical memory
  - An operating system is doing a lot of work on your behalf
  - Program #1: I want access to memory location **000000FFh** for word processing
  - Program #2: I want access to memory location **000000FFh** for web browsing
  - OS: there's only one **000000FFh** in hardware

Concept

# Memory Management

- In modern computers, the Operating System is managing the memory usage for all applications
- Memory regions are partitioned and access controlled (e.g. paged virtual memory)
  - So different programmes cannot accidentally read/write other programmes' memory
  - So different hardware controllers have easy ways of interacting with the OS

Concept

# Memory Protection

- Hardware: MMU (Memory Management Unit)
  - Early computers do not all have MMU, which is critical for mission critical usage
- Software: By the OS. OS checks restrictions with every memory read/write operation by applications/drivers etc.
  - Windows NT+; Mac OS X+; Unix/Linux
- Why is Memory Protection important: apart from privacy, so when one programme crashes, it doesn’t bring down the whole system

# MMU: ARM VMSA

- ARM Architecture uses VMSA (Virtual Memory System Architecture
  - Handles Virtual-to-Physical address mapping
  - Handles memory allocation for processes
  - Completely implemented in Hardware:
    - translation table from virtual to hardware (**TLW**) is implemented in hardware
    - this process is slow, so it's also buffered (cached) using **TLB**s (translation lookaside buffers)
  - System Control Coprocessor (CP15) registers control the VMSA, which stores memory status and fault codes, address information, etc.

1. https://developer.arm.com/documentation/ddi0406/b/System-Level-Architecture/Virtual-Memory-System-Architecture--VMSA-/About-the-VMSA

# Memory "Partitioning"

- Kernel segments
  - Invisible to the user
  - Contains OS system call code, file system and resource management stuff, driver code, etc.
- User stack
  - for user applications
  - created at runtime
- User heap
  - for user applications
  - created when user dynamically allocate memory

1. Modern OS don't typically partition physical memory, but different pages can be used for different segments to allow for maximum flexibility
2. You will learn more in an OS course

# A Programmer's Perspective

- When you write a programme, usually your code is not directly accessing physical memory
  - Program #1: I want access to memory location **000000FFh** for word processing
  - Program #2: I want access to memory location **000000FFh** for web browsing
  - OS: programme #1: you have access to page **#XXX**, **000000FFh** is translated to physical address **XXXXXXXXh** (invisible to user), all your access to **000000FFh** is directed by the OS to this address
  - OS: programme #2: you have access to page **#YYY**, **000000FFh** is translated to physical address **YYYYYYYYh** (invisible to user), all your access to **000000FFh** is directed by the OS to this address
- Programmer: **I don't care about physical addresses in my application**
- This is called accomplished using **paged virtual memory**

# Static RAM Design

# Static RAM

- SRAMs work by employing latches to store information (typically SR Latches)
- SRAM will hold information permanently unless power is cut
- SRAM uses latches, which requires more components than DRAM so more expensive, but are also much faster
- SRAMs are used in early generations of PCs as main memories, like the VIC-20[1]
- SRAMs now are often used as L1 cache, the fasted and smallest cache in CPU
    - Not as fast as registers though due to design restrictions. Register operations usually require a single cycle, L1 cache starts at x2.5 (why?)

Concept

1. https://en.wikipedia.org/wiki/Static_random-access_memory#:~:text=SRAM was used for the,100, and VIC-20.

# Static RAM

- This is a single SRAM cell
- Latches are transparent
  - But RAM doesn't at anytime perform Read/Write simultaneously, so this is less important
- `Select`: like an enabling signal
  - `Select <= 0`, outputs 0 and no inputs are accepted
  - `Select <= 1`, inputs are accepted and outputted

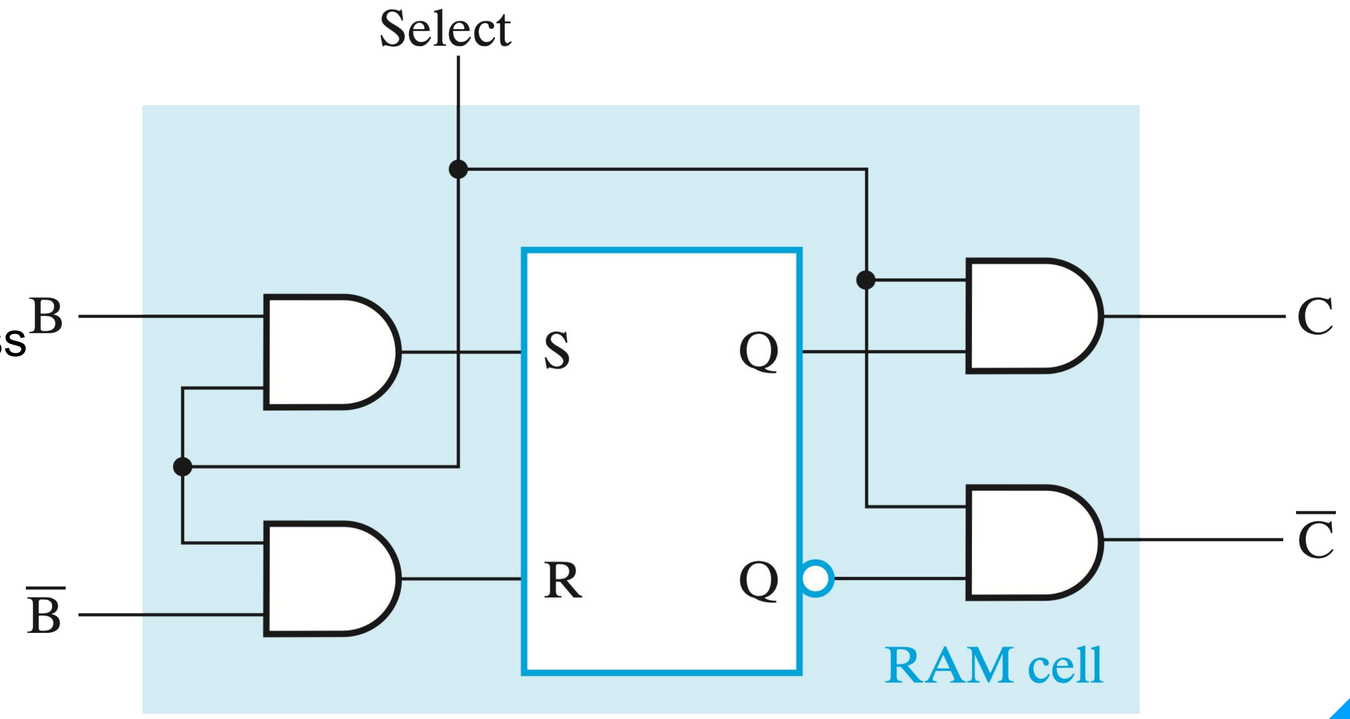
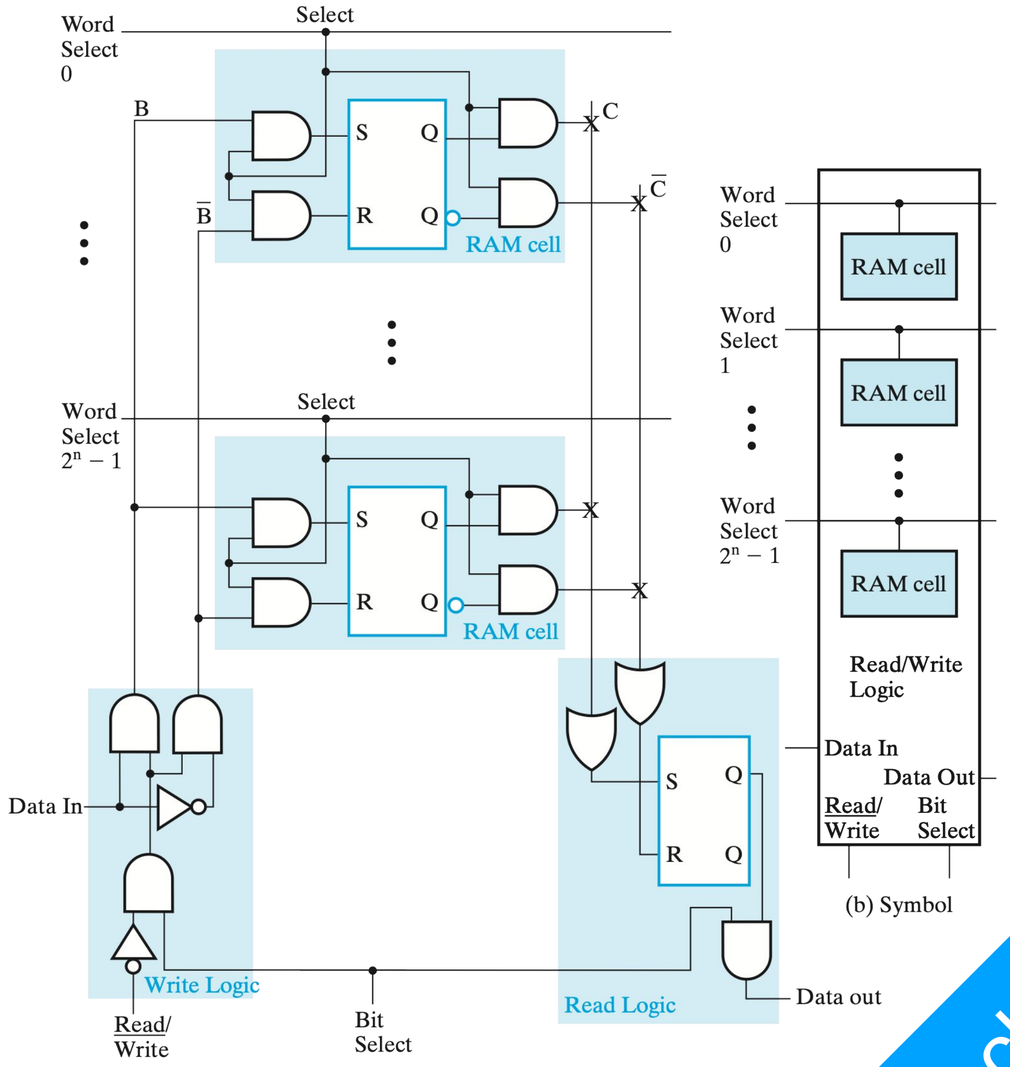



**FIGURE 7-4**
Static RAM Cell

# Static RAM

- This is the diagram for $n$-bit SRAM
- `Select` signal is shared between all bits of a single word, coming from a decoder
- `Data_in` is combined with `Read/Write` and fed into each $n$-bit SRAM cells
- `Bit_Select` is like `Chip_Select`/`Chip_Enable`, for enabling the entire circuit
- The output side are all fed into $m$-input OR gate



# Static RAM

- 16 x 1bit RAM Using a 4 x 4 RAM Cell Array
- Row selection and Column selection done separately to avoid more complex decoder designs
  - Also allows the addition of more memory cells easily by the user